Treball de Fi de Master

Master's Degree in Automatic Control and Robotics

Planning and control of an autonomous vehicle using Takagi-Sugeno and MPC techniques.

MEMÒRIA

Autor: Álvaro Herreras García

Director: Dr. Vicenç Puig Cayuela

Convocatòria: June 2018

Escola Tècnica Superior

d'Enginyeria Industrial de Barcelona

Abstract

Nowadays, many research efforts are focused on autonomous driving. It is a field that is so interesting because it joints different areas within engineering. This work is focused on the controller design part, presenting a way of designing a controller for an autonomous car.

The proposed controller is a Model Based Predictive Controller (MPC), which is a way of designing a controller based on optimization terms. Some optimization tools are used in order to get at each time instant the best control action to apply to the vehicle, taking into account the future predicted behavior of the system.

The project takes advantage of the possibility of addressing the modelling challenges that autonomous driving proposes from different ways. In this thesis, the novelty resides in the control oriented model construction, which is based on the use of the Takagi-Sugeno approach, a modern modelling technique based on fuzzy laws.

Finally, a set of tests with its corresponding results are presented and analyzed, looking for the advantages and disadvantages of proposed approach. Also, a report of the social impact the project would cause is included, and the possibility of doing it from a professional framework is explored.

Contents

1. Introduction 1

1.1 Motivation 2

1.2 Objectives 2

1.3 Outline 3

2. State of the art 5

2.1 Autonomous driving 5

2.2 Takagi-Sugeno 6

3. Vehicle modelling 8

3.1 Non-linear model 8

3.1.1 Dynamic model 9

3.1.2 Kinematic model 10

3.1.3 Control-oriented model (COM) 11

4. Controller design 20

4.1 Model Predictive Controller (MPC) 20

4.1.1 Inputs/Outputs 20

4.1.2 Objective function *J* 21

4.1.3 Constraints 22

4.1.4 Optimization problem 23

4.2 Moving Horizon Estimator (MHE) 24

5. Tuning 28

5.1 MPC tuning 28

5.2 MHE tuning 30

5.3 Matrices update 31

5.3.1 Frozen matrices 32

5.3.2 Update according to prediction 32

ETSEIB

5.3.3 Update according reference 33
6. Simulation results 34
6.1 Testing circuits 34
6.1.1 Circuit 1 34
6.1.2 Circuit 2 36
6.1.3 Circuit 3 37
6.2 Final circuit 39
7. Environmental and social impact 42
7.1 Environmental impact 42
7.2 Socioeconomic impact 42
8. Budget 44
9. Conclusions 46
9.1 Future work 47
Bibliography 48

List of figures

Figure 1.1. Electric car 2
Figure 3.1. Vehicle dynamics schema 9
Figure 4.1 MPC I/O schema 21
Figure 4.2 MHE schema 26
Figure 4.3 Control loop 28
Figure 6.1 Circuit 1 35
Figure 6.2 Circuit 1 results 35
Figure 6.3 Circuit 2 36
Figure 6.4 Circuit 2 results 37
Figure 6.5 Circuit 3 38
Figure 6.6 Circuit 3 results 38
Figure 6.7 Final circuit 40
Figure 6.8 Final circuit results 40

List of tables

Table 3.1 Vehicle parameters 9
Table 6.1 Testing circuits results..... 39
Table 6.2 Final circuit errors..... 41
Table 8.1 Project cost structure 44

1. Introduction

Autonomous vehicles are quite popular nowadays. It seems that the future tends to be a time where the intervention of the people in driving tasks will be reduced to the minimum. The improvement of this kind of transport techniques until the point that people had maximum confidence on them will suppose a change in many processes as e.g. transport, communication or deliveries.

In this work, a research on a particular control technique is presented. A particular approach is proposed to face the common challenge of autonomous cars as following a path in a way that guarantees stability and accuracy. The project is developed using Matlab, testing the results against a model of simulation that is very close to the real behavior of the car.

This thesis can be considered as developed within the context of the *Automated and Cooperative Driving in the City (ACDC)* that is a project in collaboration with the Computer Vision Centre (CVC) and UPC. The CVC is automatizing an electric car *(Figure 1.1. Electric car)* that uses a stereo rig in order to detect the obstacles and generate a path with its corresponding velocities, positions, steering angle, etc. This path is the input of our controller that has to compute in each state the corresponding control actions to make the car follow this path in a proper way.

Figure 1.1. Electric car

1.1 Motivation

The transport and the communications are two main factors directly related with progress. All tasks, from daily people ones to international relationships or commercial relationships are improved as long as the modes of transport improve. It is a fact that many research efforts nowadays are focused on autonomous driving and they will make the difference in the near future.

That is way autonomous driving is an interesting topic to focus on, and, this work is the chance to face some of its challenges, working on an electric car that makes minimum impact to the environment.

The idea of developing a safer, lower pollutant and more comfortable mode of transport, combined with the application of one modern control technique inspire this work that allows merging all these factors giving the possibility of contributing in the change that all modes of transport will have in the future.

1.2 Objectives

The main objective of this thesis is to design a controller for a particular electric car in order to follow a desired trajectory and arrive to a goal. The plan of the desired trajectory is out of the scope of this work.

To do so, a first task of model extraction is carried out, giving us a non-linear model that represents the car in our simulation tests. This task is followed by a linearization part, a process oriented to control design that mainly transforms the non-linear model of the car into a linear parameter varying model, applying fuzzy techniques (Takagi-Sugeno fuzzy models).

After that, the designing process starts, testing different approaches of a model based predicted controller (MPC) we will try to get the best sequence of control actions that makes the car follow the reference trajectory, and finally we will see which one has better performance using the simulation-oriented model.

1.3 Outline

The thesis is structured in the following way:

Chapter 2

In this section, the state of the art of autonomous driving is analyzed, studying its background in order to have a clear perspective of the situation we are today.

Chapter 3

The mathematical model of the vehicle is transformed into a linear-like one in this chapter. An introduction of Linear Parameter-Varying (LPV) and Takagi-Sugeno (TS) models is presented, followed by the application of this linearization to our non-linear model. This will provide the control-oriented model that will be used for control design purposes.

Chapter 4

In this chapter, the whole part of the controller design is described.

On a first part, using the TS model of the vehicle, the idea is to solve in each step an optimization problem that will give us the optimal control action the car requires. We will see the problems of dealing with different solvers and how they affect the results.

On the second part of this chapter, a Moving Horizon Estimator (MHE) is implemented, the objective of this observer is to estimate the states of the vehicle that are not measurable.

Chapter 5

Here, the tuning of the controller weights is described. This is an important part in the design of MPC because allow us to regulate the relation between performance and efficiency of the controller. Results obtained using different weights will be presented.

Chapter 6

This is the chapter where the results will be discussed. Taking advantage of working with a simulation, we will do several tests with different kind of controllers and with different circuits in order to have an idea of the robustness of the controller.

Chapter 7

The environmental impact and the possible repercussion of this project on the society will be analyzed within this section.

Chapter 8

This chapter is related with the possibility of integrating this work to the professional framework. An study about the possibility of developing this thesis beyond the University environment is presented.

Chapter 9

All the conclusions that can be extracted from the work done are compiled in this section, studying the proposed objectives and compare them with the results obtained.

Chapter 10

In this final chapter a possible continuation of the work is presented, establishing new objectives and adapting the ones that could not be reached in that case.

2. State of the art

2.1 Autonomous driving

It is important to contextualize our work according to some background or projects previously developed in this area, so the aim of this section is exactly addressing this goal.

In this context, we can highlight one of the first and most important research in the autonomous driving area, the first *NAVLAB* vehicles at *Carnegie Mellon University (Thorpe 1990).* They were used to develop, integrate and test advanced technologies for autonomous vehicle guidance. These vehicles could be driven in road networks, and they worked as a way to introduce concepts like occupancy grids, to improve investigations in path planning algorithms, etc.

Later on, some improvements were made in the framework of challenges, for instance, the *Defense Research Advance Projects Agency (DARPA),* planned some races that pushed the research in this field. One of these races was the first *Grand Challeng*e, which took place in 2004. It was a 142 mile off-road course with a requirement to complete it within 10 hours, but none of the participants succeeded. However, a year and a half later the second *Grand Challenge* took place, a course vary similar to the first one that five of 195 teams could finish. Finally, a third *Grand Challenge* was addressed, introducing an urban environment with traffic rules and interactions with other vehicles, and 6 out of 89 teams could finish the event. The *DARPA* challenges were so important in the autonomous driving research and introduced many advances in the field, such as knowledge on sensors or devices used or reactions between real drivers and autonomous cars.

Moving to Europe, *Cybercars* is the one of the major efforts done in order to develop fully automated vehicles moving in town centers. These developments were aimed to work in areas without other autonomous vehicles but with pedestrian traffic at very slow speeds.

Nowadays, many big companies are introducing autonomous driving techniques in their cars, beginning from “easy” tasks such as autonomous parking and evolving these techniques until the development of commercialized cars that are fully able to follow trajectories by itself at some points of the route.

2.2 Takagi-Sugeno

Around the eighties, the concept of gain-scheduling was introduced in the control and modelling areas. The basic concept was to linearize the nonlinear system at different operating points resulting in a set of Linear Time-Invariant (LTI) models of the plant (multi-models). Due to many successful applications (Spong 1987; Whatley and Pot 1984; Stein 1980), the methodology has become very popular in industrial applications. It has been proved that in general, many nonlinear systems can be converted into a linear parameter varying (LPV) form. In parallel with the development of this approach, Takagi and Sugeno studied another way to face these “multi-models”. Tthis approach is rooted in the fuzzy logic theory. The idea is to describe a non-linear systems via a set of IF-THEN rules and local models that are combined via a membership function following the so-called Takagi-Sugeno modelling approach [1].

In the past decades, there are many research works focused on Takagi-Sugeno modelling, for instance [1], presents an approximation of Takagi-Sugeno modelling in order to deal with nonlinear system subject to parameter uncertainty, and proposes a new type of state feedback controller named interval type-2 regional fuzzy to handle this systems using linear matrix inequalities (LMI’s).

The Takagi-Sugeno modelling was combined in recent years with many different control techniques. [2] presents a study of observer design techniques based on Takagi-Sugeno modelling, and [3] provides a learning algorithm of Takagi-Sugeno fuzzy models aimed to automatically extract all the parameters of a fuzzy logic system.

Regarding to the autonomous driving topic, [4] present a studio of the application of a Takagi-Sugeno fuzzy controller used for steering commands generation, in the sense of GPS based autonomous navigation of heavy vehicles at high speed. In the framework of lighter vehicles, [5] developed a novel control law for an autonomous vehicle: the goal is to perform a lane keeping under multiple constraints, namely actuator saturation of the steering system, roads with unknown curvature, and uncertain lateral wind force, to achieve it, a novel Takagi-Sugeno control method using non-quadratic Lyapunov stability framework is proposed.

This thesis is based on the precious work of [6] , which studied the planning and control techniques of the same autonomous car from the point of view of the nonlinear control

approaches. The trajectory planner developed in that work is used here, with the aim of applying the Takagi-Sugeno modelling and Model Predictive Control techniques.

As we can see, there are many research results in the field of Takagi-Sugeno modelling, a kind of modelling that allows us to apply it with different control techniques such as linear matrix inequalities or robust H_∞ control for instance.

This work poses the application of this kind of linearized systems with Model Predictive Controllers together to be able to handle nonlinear systems using a linear-like approach.

3. Vehicle modelling

In this section, the challenge of representing mathematically the vehicle is addressed. This part plays an important role in our project because the more we work in this area, the closer our simulation is to the real vehicle.

Two different models are needed in this case, the simulation-oriented model (SOM) and the control-oriented model (COM). The SOM is only used for simulation purposed and for testing the proposed approach. It is required to present the closest to reality behavior, no matter the complexity of the model. However, the COM is a model that requires to be the SOM with some transformations (simplifications), in order to make the control design task simpler in terms of the computational point of view.

3.1 Non-linear model

The vehicle model we start working with is a non-linear one, and it can be divided in two different parts, the kinematic one and the dynamic one. This model takes into account all the possible movements of the vehicle (six degrees of freedom) [6].

In *Table 3.1 Vehicle parameters*

, all the vehicles parameters are presented, and they are used in the dynamic model and the kinematic one as well. *Figure 3.1. Vehicle dynamics schema* present he schema of the behavior of the car based on the so-called bicycle model.

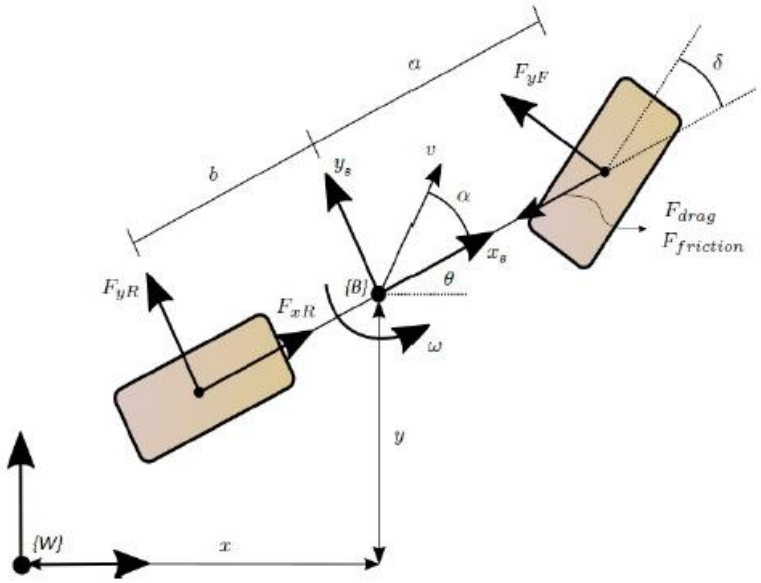

Figure 3.1. Vehicle dynamics schema

Parameter	Description	Value
a	Distance from CoG to front axle	0.758 *m*
b	Distance from CoG to rear axle	1.036 *m*
M	Vehicle mass	683 *kg*
I	Vehicle yaw inertia	560.94 *kg* m^2
C_d	Drag coefficient	0.36
A_r	Vehicle frontal area	1.91 m^2
ρ	Air density (25ºC)	1.184 $\frac{kg}{m^3}$
μ_0	Nominal friction coefficient	0.5
C_x	Tire stiffness coefficient	25000 $\frac{N}{rad}$

Table 3.1 Vehicle parameters

3.1.1 Dynamic model

The dynamic model of the vehicle that is used in this project is based on the second Newton's law, obtaining a model with three states and two inputs that is the following one:

$$\dot{v} = \frac{F_{xR}\cos(\alpha) + F_{yF}\sin(\dot{\alpha} - \delta) + F_{yR}\sin(\alpha) - F_{df}}{M}$$

$$\dot{\alpha} = \frac{-F_{xR}\sin(\alpha) + F_{yF}\cos(\dot{\alpha} - \delta) + F_{yR}\cos(\alpha)}{Mv} - \omega$$

$$\dot{\omega} = \frac{F_{yF}a\cos(\dot{\delta}) - F_{yR}b}{I}$$

Where:

$$F_{yF} = C_x(\delta - \alpha - \frac{a\omega}{v})$$

$$F_{yR} = C_x(-\alpha - \frac{b\omega}{v})$$

$$F_{dF} = F_{drag} + F_{friction} = \frac{1}{2}C_d\rho A_r v^2 + \mu_0 Mg$$

All these equations are subject to the parameters in *Table 3.1 Vehicle parameters*

. This model is represented in state-space. The state variables are: the vehicle velocity v, the slip angle α and the angular velocity ω .

As inputs, we have the steering angle δ and the longitudinal rear force F_{xR}. F_{yF} and F_{yR} represents the lateral rear and front force that appears with the angular motion. F_{drag} is the drag force opposite to the forward movement and $F_{friction}$ is the friction force, also opposite to the forward movement.

3.1.2 Kinematic model

The kinematic model assumes null skidding and it is a geometric way to get the position and orientation of the vehicle as function of linear and angular velocities.

As a result, we have a model with three states and two inputs as well

$$\dot{x} = v\,cos(\theta)$$

$$\dot{y} = v\,sin(\theta)$$

$$\dot{\theta} = \omega$$

where $x\ y\ \theta$ represent the position and orientation of the vehicle respectively, with respect to the world frame or initial frame. Later, we will see that for control purposes it is convenient to pass from the initial frame to the vehicle frame.

The inputs to this model are the linear velocity v and the angular velocity ω. Note that these two inputs are given by the dynamic model of the vehicle, so if we merge these two models together (kinematic and dynamic), we obtain the whole model of our vehicle, with six states and two inputs:

$$x = \begin{bmatrix} x \\ y \\ \theta \\ v \\ \alpha \\ \omega \end{bmatrix}; u = \begin{bmatrix} F_{xR} \\ \delta \end{bmatrix}$$

This model constitutes the simulation-oriented model (SOM), it tends to be equal as the behavior of the real car, however, for control-design it is used another model, the control-oriented model (COM) that is presented now.

3.1.3 Control-oriented model (COM)

As it was explained before, for control purposes it is convenient to make some transformations with the nonlinear model of the vehicle. In this case, and starting from the model presented in the section before, it has been decided to work with a Takagi-Sugeno approach of the nonlinear model.

3.1.3.1 Takagi-Sugeno modelling

Starting with the nonlinear model, the objective is to get a linear model with some parameter dependency. The model presented by Takagi and Sugeno is described by a fuzzy version of classical logic IF-THEN rules, representing local linear input/output relations of the nonlinear system [7].

This kind of modelling gives as a result a set of linear systems belonging each one of them to a particular IF-THEN rule, which all together represent the whole nonlinear plant.

In this case, the process is divided into two parts.

First of all, the nonlinear system (dynamic and kinematic) is transformed into a linear one with varying parameters. In the case of the kinematic model, a previous transformation is

carried out. This transformation consists on representing the model with respect to the error, and then rotating these errors in order to be represented with respect to the frame of the vehicle. So the new states of the model are:

$$x_k = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix}$$

where x_e, y_e and θ_e represents the difference between the reference and the current value. Note that the reference is given with respect to the global frame, so the rotation is done after computing the error:

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{bmatrix}$$

So, deriving and taking into account some trigonometric rules, the model is obtained

$$\dot{x_e} = \omega y_e + v_d \cos(\theta_e) - v$$

$$\dot{y_e} = -\omega x_e + v_d \sin(\theta_e)$$

$$\dot{\theta_e} = \omega_d - \omega$$

The matrices of the model are dependent on some parameters as it was said before. These parameters (scheduling variables) for the kinematic model are ω, v_d and θ_e. And the model is represented as it follows:

$$\dot{x_k} = A_k(\omega, v_d, \theta_e)x_k + B_k u_k - B_k r_k$$

where:

$$A_k(\omega, v_d, \theta_e) = \begin{bmatrix} 0 & \omega & 0 \\ -\omega & 0 & v_d \dfrac{sin\theta_e}{\theta_e} \\ 0 & 0 & 0 \end{bmatrix}$$

$$B_k = \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix}$$

$$r_k = \begin{bmatrix} v_d cos\theta_e \\ \omega_d \end{bmatrix}$$

This reference vector is only taken into account at the end of the control law. And the inputs to the system are:

$$u_k = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Now, we proceed to construct the fuzzy laws, as we have 3 scheduling variables, we will have 8 fuzzy laws (2^3). Each scheduling variable will vary along its own limits, within a minimum and a maximum value. For our scheduling variables, these intervals are:

$$\omega = [-1.4147, 1.4147] rad/s$$

$$v_d = [2, 18] m/s$$

$$\theta_e = [-0.139, 0.139] rad$$

The rules are built considering these limits for each variable. So, each rule will be associated to a subsystem which corresponds to a max/min combination of the scheduling variables, and then the resulting model will be a combination of all these rules. In the considered model, only the *A* matrix is dependent to the parameters, so each subsystem will be different form the other ones just in this matrix. The rules are the following:

IF ω*=min,* v_d*=min and* θ_e*=min*

THEN

$$\dot{x_k} = A_1 x_k + B_k u_k - B_k r_k$$

IF ω*=min,* v_d*=min and* θ_e*=max*

THEN

$$\dot{x_k} = A_2 x_k + B_k u_k - B_k r_k$$

IF ω*=min,* v_d*=max and* θ_e*=min*

THEN

$$\dot{x_k} = A_3 x_k + B_k u_k - B_k r_k$$

IF ω*=min,* v_d*=max and* θ_e*=max*

THEN

$$\dot{x_k} = A_4 x_k + B_k u_k - B_k r_k$$

IF ω*=max,* v_d*=min and* θ_e*=min*

THEN

$$\dot{x_k} = A_5 x_k + B_k u_k - B_k r_k$$

IF ω*=max,* v_d*=min and* θ_e*=max*

THEN

$$\dot{x_k} = A_6 x_k + B_k u_k - B_k r_k$$

IF ω*=max,* v_d*=max and* θ_e*=min*

THEN

$$\dot{x_k} = A_7 x_k + B_k u_k - B_k r_k$$

IF ω*=max,* v_d*=max and* θ_e*=max*

THEN

$$\dot{x_k} = A_8 x_k + B_k u_k - B_k r_k$$

This combination gives us a set of eight *A* matrices, corresponding each one of them to one vertex related with the limit of the scheduling variables. The whole system is now composed by a part of each one of the subsystems [8] depending on the value of the scheduling variables at each time instant.

In order evaluate at each time instant, the system matrices, a function is used. For each scheduling variable, two functions are defined to evaluate in a mathematical way how close we are to the maximum of the variable and another one to evaluate how close we are to the minimum. These functions are called the membership functions, and for each variable are the following ones:

$$E_{max} = \frac{\omega - \omega_{min}}{\omega_{max} - \omega_{min}}$$

$$M_{max} = \frac{v_d - v_{d\,min}}{v_{d\,max} - v_{d\,min}}$$

$$L_{max} = \frac{\theta_e - \theta_{e_{min}}}{\theta_{e_{max}} - \theta_{e_{min}}}$$

And the corresponding ones to the minimum values:

$$E_{min} = 1 - E_{max}$$

$$M_{min} = 1 - M_{max}$$

$$L_{min} = 1 - L_{max}$$

The computation of the complete system is done as a weighted sum of the subsystem matrices, and the evaluation of the membership functions (between 0 and 1) allows us to compute the vector of weights that will be multiplied by its corresponding matrix. Each weight has a matrix associated.

The weights are computed as it follows:

$$weights = \begin{bmatrix} E_{min}M_{min}L_{min} \\ E_{min}M_{min}L_{max} \\ E_{min}M_{max}L_{min} \\ E_{min}M_{max}L_{max} \\ E_{max}M_{min}L_{min} \\ E_{max}M_{min}L_{max} \\ E_{max}M_{max}L_{min} \\ E_{max}M_{max}L_{max} \end{bmatrix}$$

And the system matrix is:

$$A = \sum_{i=1}^{8} weights(i)A_i$$

As previously discussed, the matrix *A* is the only one that is dependent of the scheduling variables. So, in this case, matrix *B* does not require this computation.

Once the kinematic model is calculated, we proceed in a similar way to compute the dynamic one.

As we saw in the previous section, the dynamic model has the following states and inputs:

$$x = \begin{bmatrix} v \\ \alpha \\ \omega \end{bmatrix}; u = \begin{bmatrix} F_{xR} \\ \delta \end{bmatrix}$$

Selecting as scheduling variables δ, v and α, and making some transformation in order to have the matrices dependent to these variables we obtain the following model:

$$\dot{x_d} = A_d(\delta, v, \alpha)x_d + B_d(\delta, v, \alpha)u$$

being:

$$A_d(\delta, v, \alpha) = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{bmatrix}$$

$$A_{11} = -\frac{F_{df}}{Mv}$$

$$A_{12} = \frac{C_x(\sin(\delta)\cos(\alpha) - \sin(\alpha)\cos(\delta) - \sin(\alpha))}{M}$$

$$A_{13} = \frac{C_x(a(\sin(\delta)\cos(\alpha) - \sin(\alpha)\cos(\delta)) - b\sin(\alpha))}{Mv}$$

$$A_{22} = \frac{-C_x(\cos(\alpha)\cos(\delta) + \sin(\alpha)\sin(\delta) + \cos(\alpha))}{Mv}$$

$$A_{23} = \frac{-C_x a(\cos(\delta)\cos(\alpha) + \sin(\alpha)\sin(\delta)) + C_x \mathrm{bcos}(\alpha))}{Mv^2}$$

$$A_{32} = \frac{C_x(b - \mathrm{acos}(\delta))}{I}$$

$$A_{33} = \frac{-C_x(b^2 + a^2\cos(\delta))}{Iv}$$

$$B_d(\delta, v, \alpha) = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \\ 0 & B_{32} \end{bmatrix}$$

$$B_{11} = \frac{\cos(\alpha)}{M}$$

$$B_{12} = \frac{C_x(\sin(\delta)\cos(\alpha) - \sin(\alpha)\cos(\delta))}{M}$$

$$B_{21} = \frac{-\sin(\alpha)}{Mv}$$

$$B_{22} = \frac{C_x(\cos(\alpha)\cos(\delta) + \sin(\alpha)\sin(\delta))}{Mv}$$

$$B_{32} = \frac{C_x \cos(\delta)}{I}$$

Note that, in this case, in comparison with the kinematic case, we have parametric dependency in matrix *A* and *B*.

At this point, we continue in similar manner than with the kinematic model, as we have also three scheduling variables, we will have eight rules too. The scheduling variables vary within the next intervals:

$$\delta = [5, 55] rad$$

$$v = [2, 18] m/s$$

$$\alpha = [-0.1, 0.1] rad$$

And applying the rules to the evaluation of the matrices in the variable limits we obtain the set of rules with each corresponding subsystem:

IF δ=min, v=min and α=min

THEN

$$\dot{x_d} = A_1 x_d + B_1 u$$

IF δ=min, v=min and α=max

THEN

$$\dot{x_d} = A_2 x_d + B_2 u$$

IF δ=min, v=max and α=min

THEN

$$\dot{x_d} = A_3 x_d + B_3 u$$

IF δ=min, v=max and α=max

THEN

$$\dot{x_d} = A_4 x_d + B_4 u$$

IF δ=max, v=min and α=min

THEN

$$\dot{x_d} = A_5 x_d + B_5 u$$

IF δ=max, v=min and α=max

THEN

$$\dot{x_d} = A_6 x_d + B_6 u$$

IF δ=max, v=max and α=min

THEN

$$\dot{x_d} = A_7 x_d + B_7 u$$

IF δ=max, v=max and α=max

THEN

$$\dot{x_d} = A_8 x_d + B_8 u$$

The membership functions and the vector of weights are calculated in the same manner as we did in the kinematic case. Then, depending on the current value of the scheduling variables in each time instant, the complete system can be computed as a weighted sum of each subsystem:

$$E_{max} = \frac{\delta - \delta_{min}}{\delta_{max} - \delta_{min}}$$

$$M_{max} = \frac{v - v_{min}}{v_{max} - v_{min}}$$

$$L_{max} = \frac{\alpha - \alpha_{min}}{\alpha_{max} - \alpha_{min}}$$

$$E_{min} = 1 - E_{max}$$

$$M_{min} = 1 - M_{max}$$

$$L_{min} = 1 - L_{max}$$

$$weights = \begin{bmatrix} E_{min}M_{min}L_{min} \\ E_{min}M_{min}L_{max} \\ E_{min}M_{max}L_{min} \\ E_{min}M_{max}L_{max} \\ E_{max}M_{min}L_{min} \\ E_{max}M_{min}L_{max} \\ E_{max}M_{max}L_{min} \\ E_{max}M_{max}L_{max} \end{bmatrix}$$

$$A = \sum_{i=1}^{8} weights(i)A_i$$

$$B = \sum_{i=1}^{8} weights(i)B_i$$

The nonlinear model of the car is transformed into this linear-like model using Takagi-Sugeno approach, which is composed by a kinematic part and a dynamic part. This is the model that is used in the controller-design part that we will see later. The interconnection of both systems is represented because two states of the dynamic model correspond the input to the kinematic one. This model is not the one used in the simulation part, because the simulation model tends to be as close to the real car as possible, so this approximation is used just for control purposes.

4. Controller design

The controller is known as the system which in each step makes all the computations in order to get the input that will drive the main system according our demands. There are several ways to implement a controller, but in this work a Model Based Predictive Controller (MPC) will be considered.

The controller we are going to design is a state feedback controller. This means that computes the input to apply, from the measured states of the car, as it was said before, only five out of six states of the car can be measured. This issue is solved using a state estimator, in particular, a Moving Horizon Estimator (MHE). The challenge of including this observer is addressed in this chapter as well.

4.1 Model Predictive Controller (MPC)

In MPC, the classical control problem is transformed into an optimization problem with some constraints, then it is solved in the optimization field and finally the results are transferred back to the control area. This is the main idea of how an MPC works.

It is called predictive because it takes into account the behavior of the system along a period of time in the future, this period of time is called *prediction horizon (Hp)*. The controller computes the sequence of inputs that will minimize a function during the prediction horizon subject to some constraints.

4.1.1 Inputs/Outputs

In our case, the MPC controller needs as inputs the initial state x_{ini} that corresponds the current state of the system, the reference trajectory, which is needed in order to compute the difference between the desired state and the estimated one (error), and a set of system matrices during all the prediction horizon. As we saw before, the model of our system is changing, so each step during the prediction horizon is related with a set of matrices, and all these matrices of the model are given to the MPC as input.

As outputs we get from the controller all the sequence of optimal u and the corresponding predicted states if these control actions would be applied. Once we have all the sequence

of optimal control actions, only the first one will be applied to the system, the remaining ones will be discarded, and at the next iteration this procedure is repeated after applying the receding horizon principle.

In *Figure 4.1 MPC I/O schema* a schema of the inputs and outputs of the MPC can be seen:

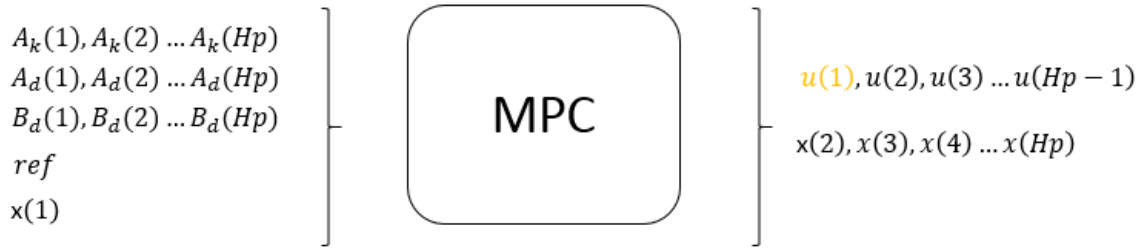

Figure 4.1 MPC I/O schema

4.1.2 Objective function *J*

The function we want to minimize is called *objective function (J),* and it has to be designed adapted to each particular control problem. There are some common used objective functions (linear, quadratic…) that penalize the error of the system and in some way usage of the actuators.

In our case, we select a function that penalizes the position error with respect to the vehicle's frame (X and Y) and the velocity error.

$$errors(k) = \begin{bmatrix} x_d(k) - x(k) \\ y_d(k) - y(k) \\ v_d(k) - v(k) \end{bmatrix}$$

Note that the desired position and velocities are given by the planner in terms of the initial global frame. So, as we did building our model, we have to rotate the errors to get them with respect to the frame of the car. Then, the errors to penalize will be:

$$errors(k) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d(k) - x(k) \\ y_d(k) - y(k) \\ v_d(k) - v(k) \end{bmatrix}$$

The way chosen to penalize this error is by using the 2-norm. There are many ways to do this and the choice of one or another depends on each one of ourselves. The part of the objective function related with the error is:

$$\sum_{k=1}^{Hp} \|errors(k)\|_2^2$$

On the other hand, if we just include the error in *J,* we let the system to use the actuators as much as possible. Thus, it is convenient to include a second element in this objective function in order to control this actuators usage in some way. A second term that corresponds to the variation of the control action Δu is included. The MPC computes the sequence of control actions during the prediction horizon *Hp,* each control action gives us a state response, and these state responses go until *Hp,* so the last control action corresponds to the instant *Hp-1.* That is way the sum of the second part of *J* ends at this point. As we did with the error, the way of penalizing this Δu is using the 2-norm

$$\sum_{k=1}^{Hp-1} \|\Delta u(k)\|_2^2$$

In addition, it is needed a way of control the priority of each term of the objective function during the optimization stage. This is done by using two matrices *Q* and *R.* These two matrices are multiplying the errors and control action respectively in the objective function. We have three errors to minimize so the size of *Q* should be 3x3, allowing us to control the minimization of each one of the errors with respect to the others as well. The same happens with the control actions, as we have two, *R* should be 2x2 and the minimization of each of them is possible because of this matrix *R.*

So finally, the objective function *J* we want to minimize is:

$$J = \sum_{k=1}^{Hp} \|Q * errors(k)\|_2^2 + \sum_{k=1}^{Hp-1} \|R * \Delta u(k)\|_2^2$$

Note that the larger *Q* with respect to *R*, the smaller the errorwill have and the larger *R* with respect to *Q*, the less energy we will spend using the actuators. The choice of *Q* and *R* is a hard task that is explained in Chapter 5.

4.1.3 Constraints

Once we have the function we want to minimize *J,* and the variable that intervenes in the optimization *u,* we have to set the constraints that have to be satisfied during the optimization loop.

There are two groups of constraints: the physical limits of the variables, and the evolution of the variables (model).

As variable limits we have:

$$\begin{pmatrix} -150 \\ -0.003 \end{pmatrix} \leq \Delta u \leq \begin{pmatrix} 150 \\ 0.003 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ -0.5236 \end{pmatrix} \leq u \leq \begin{pmatrix} 7000 \\ 0.5236 \end{pmatrix}$$

The second set of constraints is formed by the evolution of the states, they are represented as the model of the plant:

$$\dot{x_k} = A_k x_k + B_k u_k - B_k r_k$$

$$\dot{x_d} = A_d x_d + B_d u$$

The model used is the control-oriented model, the TS approach of the non-linear system of the car. Obviously, this fact includes some error that is the difference between the approach and the real model, but in terms of optimization, dealing with a nonlinear model has a huge cost in terms of the computational point of view.

Note that at each time instant $A_k, A_d \; and \; B_d$ have to be computed by using the TS interpolation presented in the previous chapter.

4.1.4 Optimization problem

Merging all these previous components together the optimization problem is established as follows:

$$\min_u J$$

$$st: \dot{x_k} = A_k x_k + B_k u_k - B_k r_k$$

$$\dot{x_d} = A_d x_d + B_d u$$

$$\Delta u_{min} \leq \Delta u \leq \Delta u_{max}$$

$$u_{min} \leq u \leq u_{max}$$

where

$$J = \sum_{k=1}^{Hp} \|Q * errors(k)\|_2^2 + \sum_{k=1}^{Hp-1} \|R * \Delta u(k)\|_2^2$$

$$\Delta u_{min} = \begin{pmatrix} -150 \\ -0.003 \end{pmatrix}$$

$$\Delta u_{max} = \begin{pmatrix} 150 \\ 0.003 \end{pmatrix}$$

$$u_{min} = \begin{pmatrix} 0 \\ -0.5236 \end{pmatrix}$$

$$u_{max} = \begin{pmatrix} 7000 \\ 0.5236 \end{pmatrix}$$

This optimization problem needs to be expressed in a way such that a solver can understand it. However, we use YALMIP that is a powerful tool that transforms the problem in terms of control to the optimization point of view. This allow us to save a lot of time.

At each iteration, this problem is solved, giving us the optimal sequence of control actions *u(k)*. All of these control actions are discarded except the first one, that is the one applied to the system. At the next step, the rejected control actions will be computed again, including one more. This requires a good solver in order to be solved within a feasible time interval. In this work, we use *gurobi* as solver. The choice of the prediction horizon has to accomplish the following rules:

It has to be larger enough that the dynamics of the system are represented within the horizon, but it also cannot be too large that the computation of the optimization problem increases and makes the behavior slower. In this work, a prediction horizon of 40 samples has been chosen, it means in terms of continuous time an horizon of 2 seconds being our sampling period *Ts = 0.05 s.*

4.2 Moving Horizon Estimator (MHE)

Our model is composed by 6 state variables:

$$x = \begin{bmatrix} x \\ y \\ \theta \\ v \\ \alpha \\ \omega \end{bmatrix}$$

Regarding the state variables, the slip angle (α) is not possible to be measured. Despite we are not tracking the error of the slip angle, it is a scheduling variable regarding the TS model of the dynamic system, which means that in each step we need to know its value in order to make the interpolation of the matrices and get the current matrices of the dynamic part of the model. This is the reason why we need an estimator. Furthermore, the estimator is useful when estimating the other variables, because it gives us the value filtering the noise, so we can take advantage of that. In this work a Moving Horizon Estimator (MHE) is implemented in order to solve this problem.

MHE uses a set of previous measurements in order to obtain an estimated value of the current state, solving an optimization problem. The process is similar to the one followed by MPC but with some differences.

Instead of computing a sequence of optimal values of the optimization variable in the future as MPC does, MHE uses a sequence of past values to compute optimal current value of the optimization variable. The length of the past values taken into account is called the *observation horizon N* and it is set to be equal to the prediction horizon used in MPC.

In general terms, it uses the historical behavior of the system in order to estimate the current value. The time horizion in the past that is where we take into consideration the inputs and outputs is called the *estimation window.* This window is moving at each step, this means that the second value estimated in the previous step corresponds the initial value at this step, and this value will be forgotten in the next step, that is way is called *Moving Horizon Estimator.*

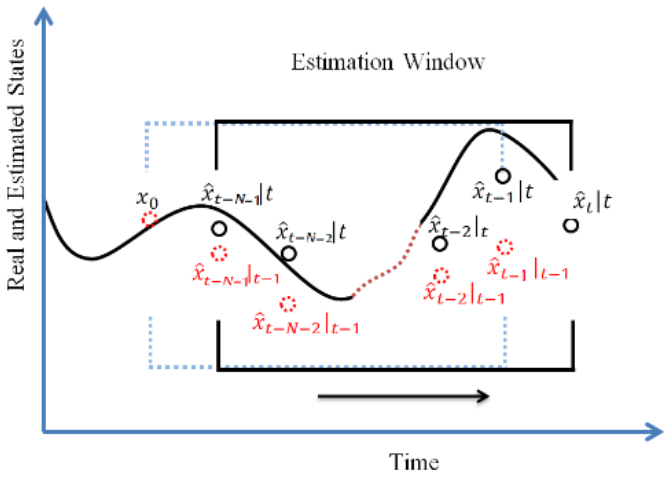

Figure 4.2 MHE schema

A set of inputs and outputs (measured) that have been applied to the system within the estimation window in the past are used to build the objective function for this optimization problem. This function contains three terms:

- The difference between the estimated state and the estate resulting from evolving the previously state estimated according the model equations:

$$\hat{x}_{k+1} - (A\hat{x}_k + Bu_k)$$

- The difference between the real output and the resulting one from the computation with the estimated state according with the model equations:

$$y_k - (C\hat{x}_k + Du_k)$$

- The difference between the first estimated state and the initial state:

$$\hat{x}_1 - x_{ini}$$

Using each of these terms, the objective function is built as follows:

$$J = \sum_{k=1}^{N-1} ((\hat{x}_{k+1} - (A\hat{x}_k + Bu_k))'Q(\hat{x}_{k+1} - (A\hat{x}_k + Bu_k)) + \big(y_k - (C\hat{x}_k + Du_k)\big)'R\big(y_k - (C\hat{x}_k + Du_k)\big) + (\hat{x}_1 - x_{ini})'P(\hat{x}_1 - x_{ini}))$$

where:

A and *B* are the system matrices (computed by the weighted interpolation each step), *D* is the output matrix, it appears if the output of the system depends directly on the input (in our case it is zero matrix), *C* is the sensor matrix, it represents the states that we can

measure. In this case, we are using MHE over the dynamic system and we are just able to measure the velocities, so *C* will be:

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

y_k and u_k are the measured outputs and inputs respectively that have been applied to the system during the observation horizon, x_{ini} is the initial value of the state, which will be the second predicted state in the step before, and $\hat{x}_k$ is the sequence of estimated states.

As we did in the MPC, we include these matrices of weights that allow us to give more priority to one or other term during the optimization process.

The optimization problem that is solved at each iteration is the following one:

$$\min_{\hat{x}} J$$

Solving this problem, we obtain an estimation of the states of the dynamic system filtering the noise, so for practical issues it is convenient to use the estimated states over the measured ones.

However, MHE is estimating just the current state when we use MPC we do not just need to know the model at the point we are but also how it will be in the prediction horizon. In the following chapter this issue will be explained in more detail.

As a summary we will recover the information related with the control and simulation loop of our system:

At the first place a planner gives us the set of references that the car needs in order to follow the trajectory.

The reference and the output of the MHE (estimated states) are introduced as inputs to the MPC, which computes the control action needed to track the reference using the control oriented model. This output is applied (just the first one) to the simulation oriented model, which is representing the real car.

The output of the simulation oriented model is used with the set of inputs and outputs stored in the past to compose the input to the MHE that estimates the current state of the car. This estimation allows to compute the current interpolation of the model matrices in an accurate way, and these matrices are the ones used by the MPC.

In *Figure 4.3 Control loop* the control schema of the vehicle can be seen.

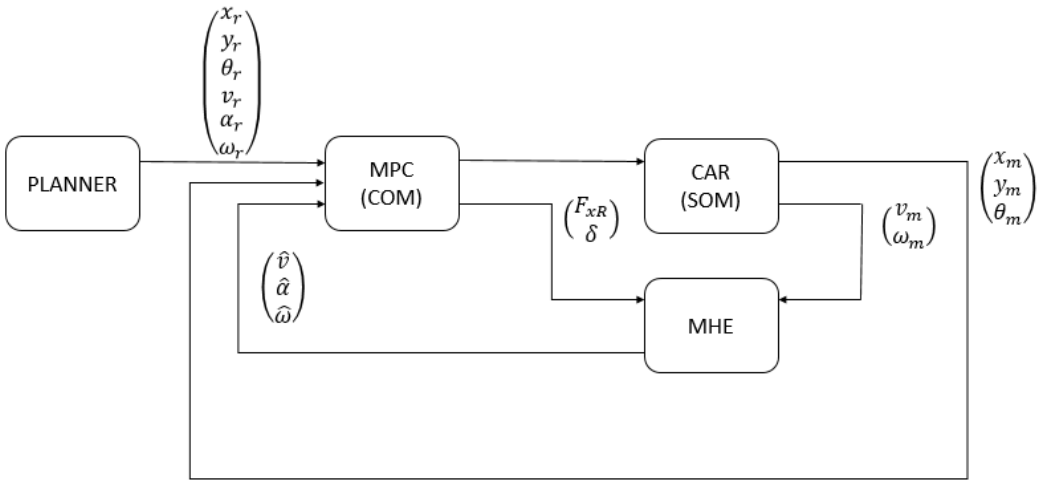

Figure 4.3 Control loop

5. Tuning

At this point, we have all the components needed to start our tests. However, the right choice of the parameters of the controller and observer is crucial factor that has a big weight on the result we will obtain.

The chapter is structured in three sections: MPC tuning, MHE tuning and Model matrices update.

5.1 MPC tuning

As it was preseented in Chapter 4, the MPC solves an optimization problem such that:

$$\min_{u} \sum_{k=1}^{Hp} \|Q * errors(k)\|_2^2 + \sum_{k=1}^{Hp-1} \|R * \Delta u(k)\|_2^2$$

$$st: \dot{x_k} = A_k x_k + B_k u_k - B_k r_k$$

$$\dot{x_d} = A_d x_d + B_d u$$

$$\Delta u_{min} \leq \Delta u \leq \Delta u_{max}$$

$$u_{min} \leq u \leq u_{max}$$

The tuning task of the MPC consists on selecting the right value of *Q* and *R* such that the error is small enough and the usage of the actuators is as less as possible. We have to considerate three elements in *Q* and two elements in *R* because we have three errors and two inputs.

In this case this tuning has been done by trial and error. The specific values of the elements is the only a way to give more priority to one term or another. So, these values are given with respect to the other ones. It is important to know that the variables we have in the optimization problem may share their units or not. So, the relation between *Q* and *R* may be clear or not.

This means that if we are working with the same units in all of the terms, if *Q=2R* it is clear that we are given twice penalization to the error over the control action, but if the control action and the error have different units this is no longer true.

In our model the variables and their units are the following:

$$errors = \begin{bmatrix} x_d(k) - x(k) \\ y_d(k) - y(k) \\ v_d(k) - v(k) \end{bmatrix} \begin{matrix} m \\ m \\ m/s \end{matrix}$$

$$\Delta u = \begin{bmatrix} F_{xR}(k) - F_{xR}(k-1) \\ \delta(k) - \delta(k-1) \end{bmatrix} \begin{matrix} rad \\ N \end{matrix}$$

As we can see the units differs ones from others, so the values of *Q* and *R* do not represent exactly how much we are penalizing one term with respect to others. Even inside each matrix we have different units so being the values of each matrix equal does not mean that the penalization of each particular error or control action will be the same.

We started setting *R=0,* in order to see if the system is able to follow the proposed trajectory without taking into account if the actuators are so used or not. This strategy allowed determining the weights inside *Q* matrix.

So, selecting *Q=diag(1,1,1)* and *R=0,* we could see that the car could not manage to follow the trajectory, at least for some critical intervals. After playing with these parameters, it was tried to arrive to a robust solution that could be valid for different circuits in simulation, because we found that some weights that allow the car a good performance could not be transferred into another circuit. When the value of *Q* was fixed, the values of *R* were increased until the overall error was interfered a point that was considered feasible.

Finally, the two matrices are set as follows:

$$Q = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R = \begin{pmatrix} 0.003 & 0 \\ 0 & 0.035 \end{pmatrix}$$

According with our definition of the errors, these weights are penalizing more the lateral error of the vehicle (y_e) than the longitudinal error (x_e). This is because we consider more important the fact that the car is able to stay aligned with the road than the fact of being exactly at the point of the road each time instant.

5.2 MHE tuning

The observer used to estimate the dynamic states of the vehicle works in a similar way as the MPC. This means that an optimization problem has to be solved as well and its corresponding weights have to be tuned. However, the variation of these weights does not have the impact in the behavior as the MPC weights have.

The optimization problem to be solved each step is the following:

$$\min_{\hat{x}} J$$

where

$$J = \sum_{k=1}^{N-1} ((\hat{x}_{k+1} - (A\hat{x}_k + Bu_k))' Q (\hat{x}_{k+1} - (A\hat{x}_k + Bu_k)) + \big(y_k - (C\hat{x}_k + Du_k)\big)' R \big(y_k - (C\hat{x}_k + Du_k)\big) + (\hat{x}_1 - x_{ini})' P (\hat{x}_1 - x_{ini}))$$

Regarding to the objective function, we can see that in this case, three matrices of weights has to be tuned, *Q, R* and *P.*

As we did in the MPC case, we tune these parameters by trial and error. The fact of working with a simulation allows measuring all the states, and this gives us the possibility of compare the results of the estimation in the current instant and the results of the real state “measured” from the simulation. In this study, no measurement noise is included.

Regarding to the difference of the real state and the estimated one is the way we can change and see how each variation of the weights affects the estimation.

After some trials, the weights used are:

$$Q = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

$$R = \begin{pmatrix} 1/30 & 0 \\ 0 & 1/30 \end{pmatrix}$$

$$P = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

5.3 Matrices update

As we explained, the matrices of our control-oriented model vary with the time depending on the scheduling variables. We have explained how to deal with this in previous chapters. We know that each time instant the majority of the states can be measured and the one that cannot be measured is estimated using MHE. However an additional problem arises.

MHE uses the information of the past to estimate the state in the present, but MPC predicts the behavior of the system along a time in the future, an along this time the model of our system is changing. This means that we do not have observer estimation of the scheduling variables during the prediction horizon of the MPC.

This problem is addressed from different perspectives:

5.3.1 Frozen matrices

One possibility is to estimate the current states using MHE, and use this information in order to make the interpolation of the matrices of the system at this time instant. Then, during the prediction horizon the model is considered constant, and an error is included during this horizon.

The optimization problem that MPC solves becomes:

$$\min_u J$$

$$st: \dot{x_k} = A_k x_k + B_k u_k - B_k r_k$$

$$\dot{x_d} = A_d x_d + B_d u$$

$$\Delta u_{min} \leq \Delta u \leq \Delta u_{max}$$

$$u_{min} \leq u \leq u_{max}$$

where:

$$A_k(1) = A_k(2) = A_k(3) = \cdots A_k(Hp)$$

$$A_d(1) = A_d(2) = A_d(3) = \cdots A_d(Hp)$$

$$B_d(1) = B_d(2) = B_d(3) = \cdots B_d(Hp)$$

This is the easiest way to proceed and obviously is not the best performance we can obtain.

5.3.2 Update according to prediction

Another way to face this problem is to update the system matrices according the predicted values provided the MPC.

As we know, MPC evolves the system in the future giving an optimal sequence of inputs that minimizes the objective function. However, we can use all the states that are resulting of the evolution of the system during the optimization states (predicted states) to evaluate the scheduling variables each time instant and make the interpolation.

This technique is more complex than the previous one, but we take into consideration the variation of the model during the prediction horizon. We decrease the error but it is not eliminated at all because we do not have real information of how the system will be.

5.3.3 Update according reference

Another possible strategy to evaluate the scheduling variables during the prediction horizon is based on update the matrices according the values of the reference.

The objective of our controller is to follow the reference, making the error as less as possible. So, if we assume that we will not have error, we can take the values of the reference from the current instant until a distance in the future equal to the prediction horizon.

The different proposed techniques have been tested and the differences between them are minimal, overall between the second and the third one. Finally, we realized that maybe the behavior is a little bit better using the reference in order to update the scheduling variables within the prediction horizon and interpolate the model matrices. However, some values (like α or δ) are not given by the planner and are taken from the prediction.

6. Simulation results

In this chapter, all the results obtained from different tests will be presented and discussed. To do that, the performance is tested over three different small circuits and finally it is tested in a big one. In all the circuits, the blue point means the initial point of the car and the red point the final one.

The model of the car used for simulation (SOM) is the non-linear model, the closest one to the real behavior of the car no matter the complexity.

The simulation procedure is the following one:

- The planner gives us a trajectory that is computed offline.
- The trajectory is passed as input to the MPC by means of the reference signal.
- Each step (Ts = 0.05 s) the controller takes the difference between the current position and the reference position (and velocity) and rotates (using the rotation matrix) these errors to the vehicle's frame.
- The controller gives as outputs the force and steering angles (control actions) that must be applied to the car to follow the trajectory.
- The control actions are applied to the SOM using ode solver of MatLab, which integrates the differential equations during a short period of time.

6.1 Testing circuits

6.1.1 Circuit 1

For the first circuit, it was chosen a simple one, a straight line with a curve at the end as the one represented in *Figure 6.1 Circuit 1*:

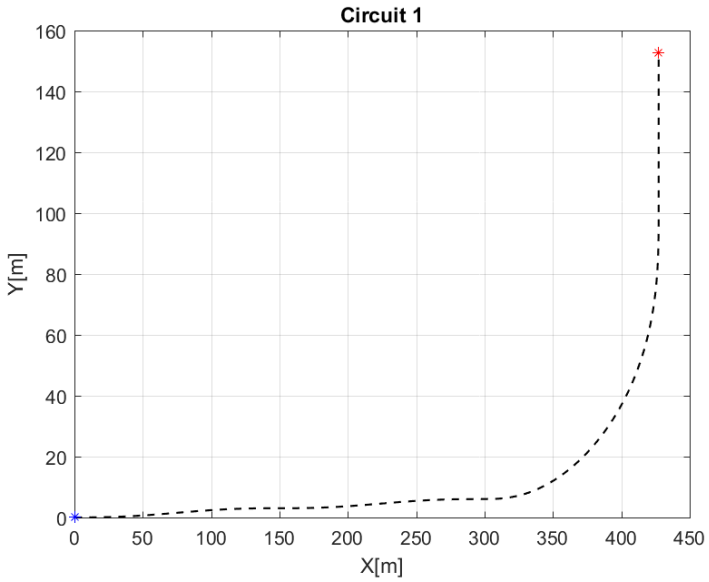

Figure 6.1 Circuit 1

All the results correspond to the weights for the MPC and MHE that were presented in Chapter 5.

For this first circuit the results obtained are the following *(Figure 6.2 Circuit 1 results):*

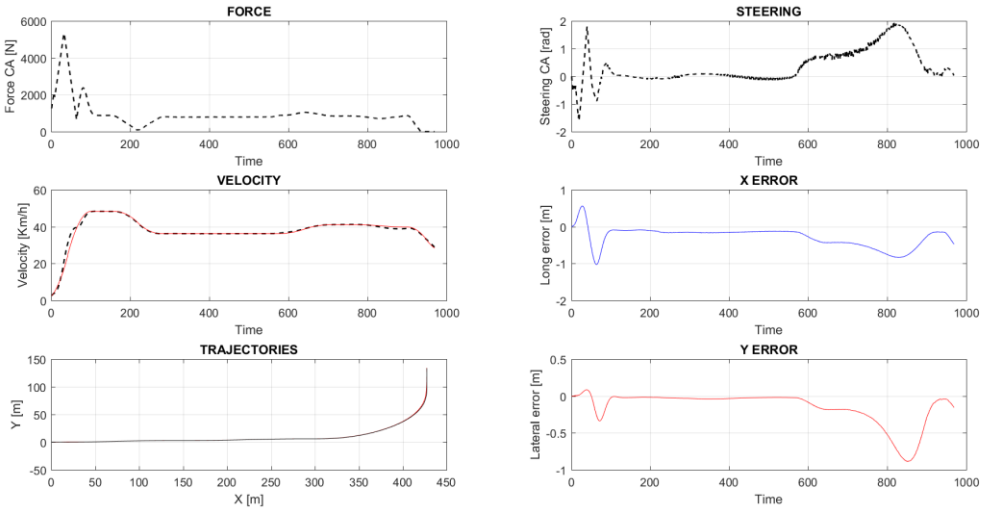

Figure 6.2 Circuit 1 results

We obtained for this first circuit a mean square error in *x* of 0.384 m and in *y* of 0.283.

Regarding to the behavior, we can see that the car can follow the trajectory in a good way in the part of the straight line but at the moment when it changes to the curve some error arises.

6.1.2 Circuit 2

The second circuit that it tested, corresponds to a movement in the other direction, a circuit that forces the car to move in both directions *x* and *y* and with more variation on the velocity reference than in the previous case.

Regarding to the control actions it is obvious to see when the variation of the velocity reference and the curve occurs, being the force and the steering maximum values respectively for both cases.

The second circuit is the following one *(Figure 6.3 Circuit 2):*

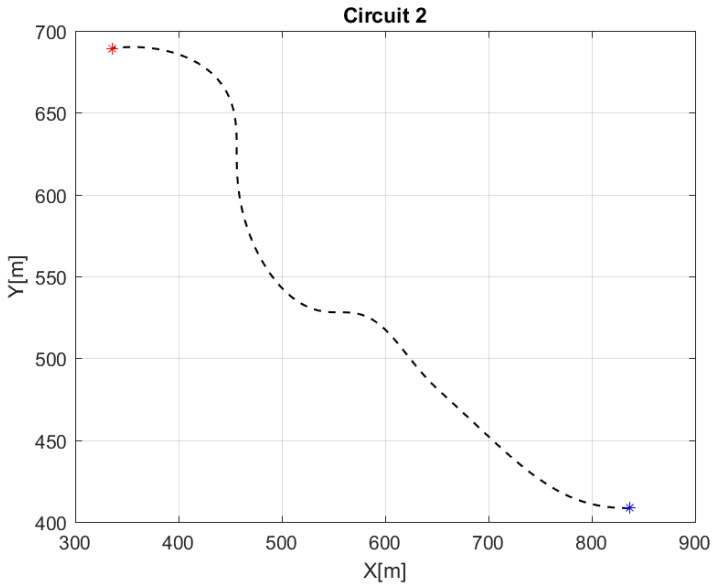

Figure 6.3 Circuit 2

This circuit is tested with the same weights as before. Note that if we tune the weights for each circuit we obtain so much better performance, but robust weights valid for all circuits were chosen. The results are the following ones *(Figure 6.4 Circuit 2 results):*

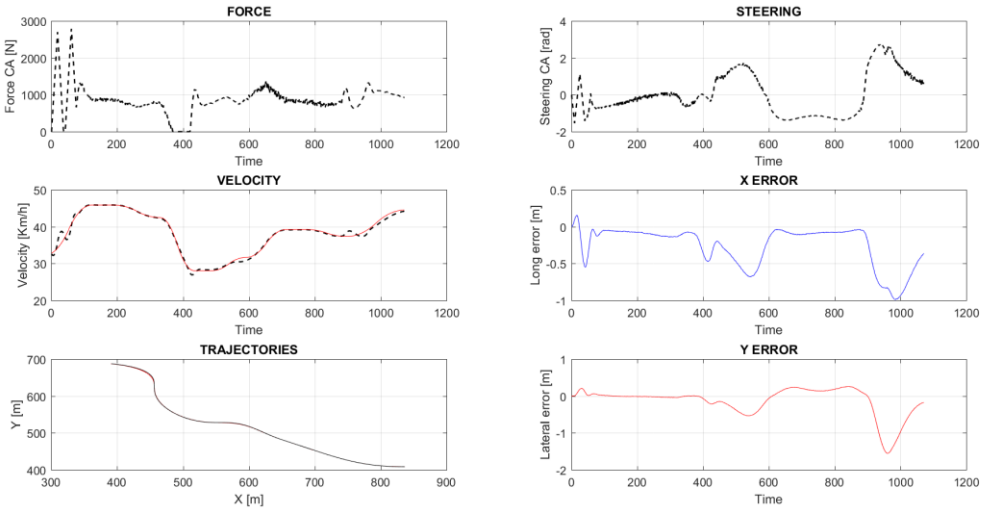

Figure 6.4 Circuit 2 results

For this test, we obtain the maximum *x* and *y* errors when changes in the velocity are introduced. This could be because there exists a disturbance that is not taken into account, as e.g. unknown friction force.

For this simulation the mean square errors obtained for *x* and *y* are 0.3863 and 0.4052 respectively.

In this case, when the reference velocity decreases we find that the car has some errors in x and y, and the controller reacts giving a maximum value for the steering control action.

6.1.3 Circuit 3

The last circuit selected to test our controller corresponds to a close curve, in order to see how the car behaves when it has to deal with this.

The curve is the following one *(Figure 6.5 Circuit 3):*

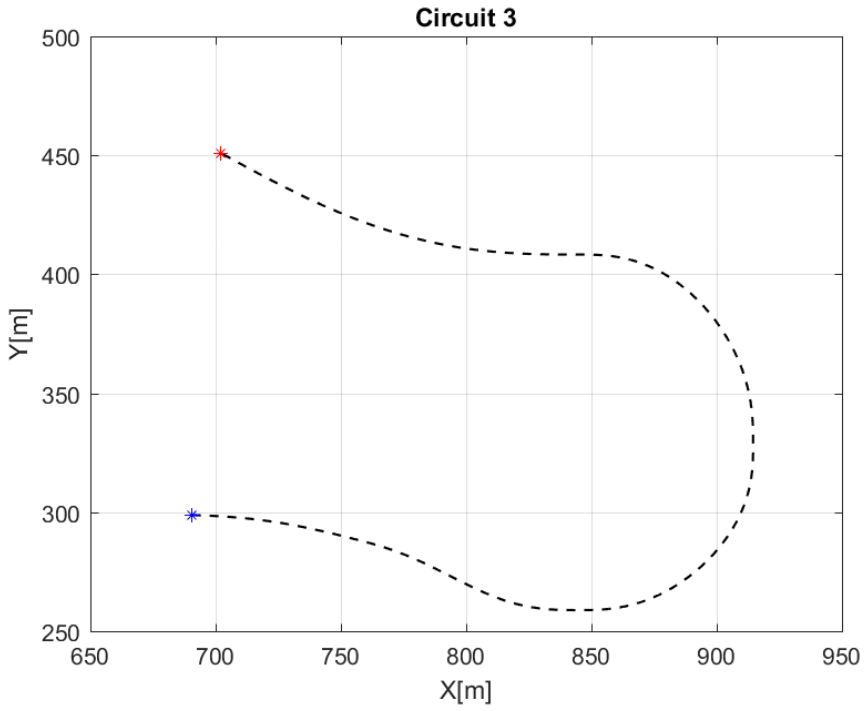

Figure 6.5 Circuit 3

And the results obtained during this trajectory *(Figure 6.6 Circuit 3 results):*

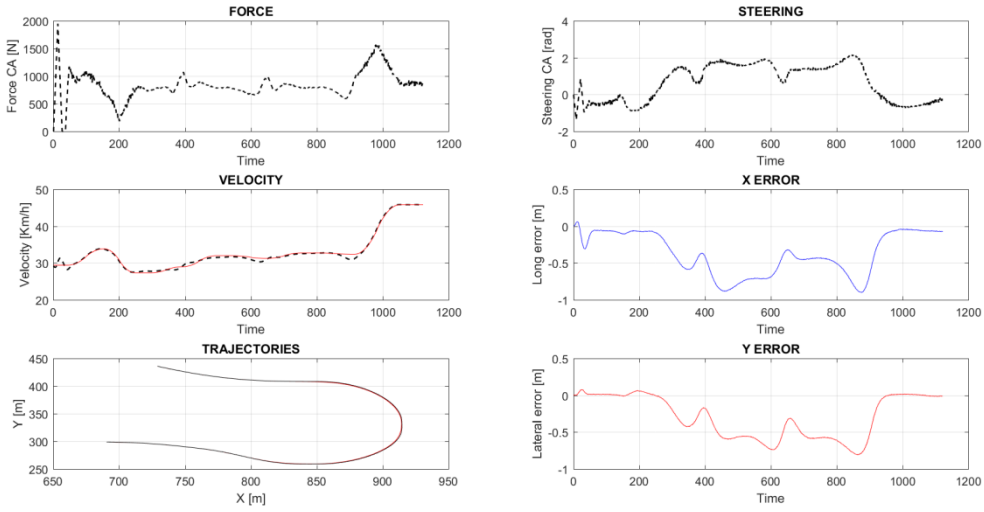

Figure 6.6 Circuit 3 results

During this circuit, we have a curve, so we can see that the steering control actions keep in a high value in order to make it. The velocity is almost the whole trajectory between 25 and

35 Km/h but in the last part it increases so it is this point when the force control action becomes maxima.

For this experiment, we obtain a mean square error in *x* of 0.4853 and in *y* of 0.4277.

Table 6.1 Testing circuits results is included as a summary of the results and circuits tested:

Circuit	MSE (x)	MSE (y)
1	0.384	0.283
2	0.3863	0.4052
3	0.4853	0.4277
$Q = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$R = \begin{pmatrix} 0.003 & 0 \\ 0 & 0.035 \end{pmatrix}$	

Table 6.1 Testing circuits results

6.2 Final circuit

A circuit was select that contains some mixed elements of the ones presented in the last section. So the car must start in one direction and shortly it should turn within a curve and go in the opposite direction, then a straight line must be follow with changes in the velocity. At the end, it must make some small curves until arrive to the destination.

The circuit is presented in *Figure 6.7 Final circuit*, being the blue point the starting and the red one the ending:

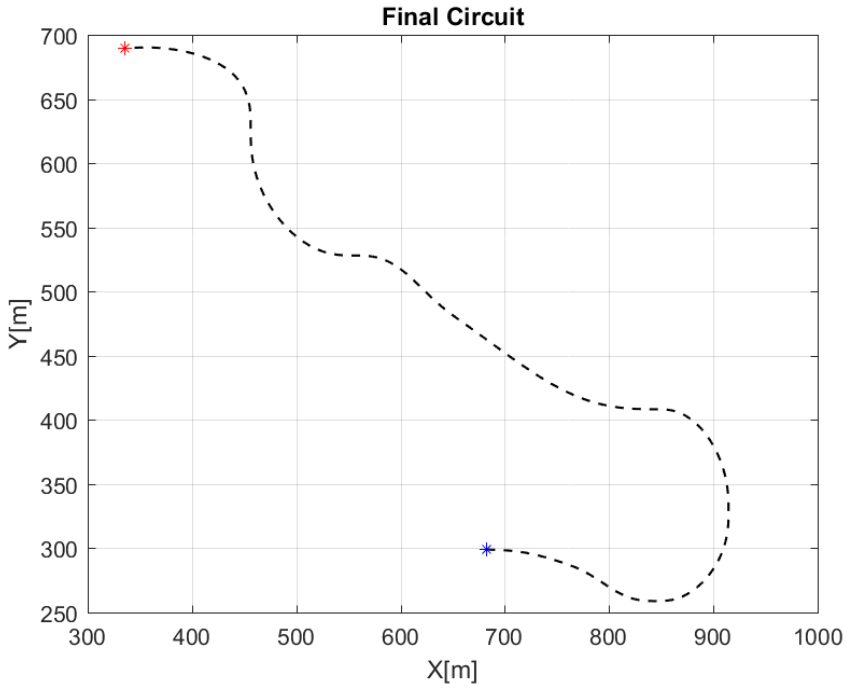

Figure 6.7 Final circuit

The objective of this circuit is to represent a possible trajectory that the real car could have to follow in order to get to the destination. The traveled distance is long enough and the circuits includes the elements needed to test the robustness of the weights.

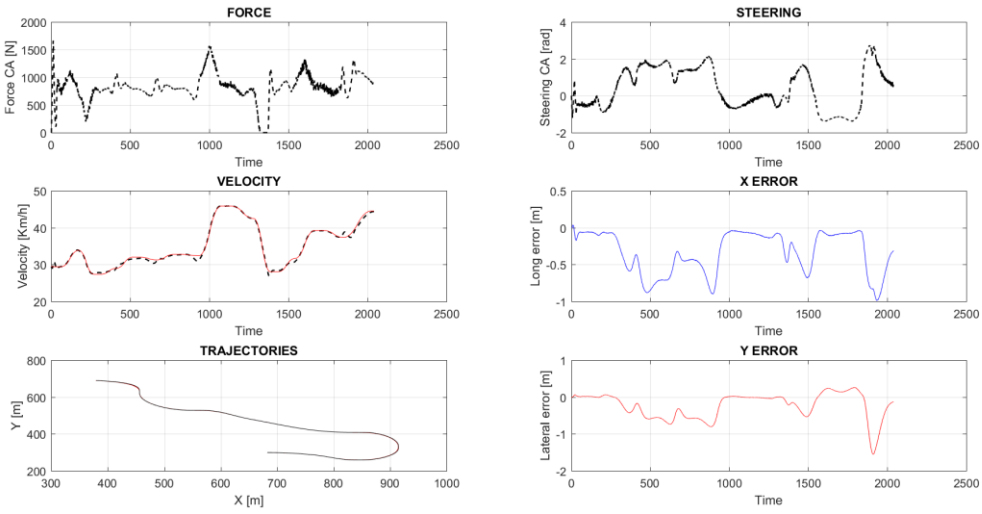

Figure 6.8 Final circuit results

In *Figure 6.8 Final circuit results* the results obtained can be seen. Regarding the trajectory, we can see that the car has been able to follow it and has been stable during all the circuit.

In the velocity graph, it can be seen how the desired velocity has been followed in a proper way, and the force control action has been able to track this reference.

Regarding to the x and y errors, we can found just a peak in the *y* error that is below -1. This peak corresponds to the final stage of the trajectory when the car is requested to make some curves while it is increasing the velocity. In the rest of the simulation, it is oscillating between [0,-1] m in the case of *x* and [0.2,-0.8] m in the case of *y*.

In *Table 6.2 Final circuit errors* we can see the errors we had:

Circuit	MSE (x)	MSE (y)	Max Error (x)	Max Error (y)
Final	0.4386	0.4277	0.9903	1.5536

Table 6.2 Final circuit errors

The max errors are given in absolute terms, note that the error in *x* and *y* is almost always below 0, that means that maybe they are due to the unknown friction force (disturbance) that is not taken into account.

7. Environmental and social impact

When some ideas are developed, and some projects are carried out, it is important to take some time to stop and think about how that idea or that project may affect the society or the environment.

In this section, a brief analysis of these aspects is presented.

7.1 Environmental impact

All of us know that conventional cars are one of the biggest sources of pollution in our society. Since some years ago, we are looking for different alternatives to petrol, such as gas or electricity.

Autonomous electric cars are one of the alternatives to this kind of pollution. It is clear that the car industry is going in that way such that many cars nowadays has an electric engine and the combustion one (hybrid cars). Moreover, the laws and legislation are trying to reduce the conventional cars utilization.

The fact of being autonomous is also an advantage. If the routes of the vehicles are automatized, the number of cars in the streets will be reduced, and transport tasks could be done in an optimal way.

The main disadvantage of including these vehicles resides in the batteries they use. The pollution that a battery for an electric car generates is linked with the improvement of the renewable energies. During the building process of the electric car's batteries half of the total pollution produced by the car takes place. However, this pollution is compensated within the two first years of usage.

7.2 Socioeconomic impact

The normalization of these kind of vehicles would suppose real big changes in society. Daily transport of people could be safer and more comfortable if the control strategies are improved and human errors are eliminated.

Not only big companies but also whole countries are dependent on petrol nowadays, the enhancement of electric cars will suppose the fall of these companies and countries. This fact could have an unpredictable impact in our society, lot of international relationships would be affected.

Furthermore, if this technology is included to transport tasks, many job positions will be lost, there are many kinds of job that are related with driving, and the whole society will be affected in some way.

In conclusion, the increase of electric cars would produce big changes in our lives, with its advantages and disadvantages and some repercussions that could be unpredictable.

8. Budget

The aim of this chapter is to provide a brief economic analysis in the case of introducing our work to the commercial reality.

This is a research work, so we will put ourselves in the place of being asked to equip some electric cars with the system tested. The idea is to give an economic value per car equipped and to do that we will suppose that we are asked to equip a total number of 200000 car units.

All the costs that are taken into account are presented in *Table 8.1 Project cost structure*

Description	**Estimated cost (€)**
Computational Device (cd)	650 (u)
Sensors (sen)	1850 (u)
Other autom. Devices (comm., act…) $(adev)$	930 (u)
Equipment labor (el)	300 (u)
Algorithm development (ad)	375000
Software licenses (MatLab, Graphical Env…) (sl)	2500
Maintenance (technical support) $(maint)$	2720
Simulation tests $(simt)$	880
Other costs (taxes, development devices, electricity…) (oc)	2300

Table 8.1 Project cost structure

The costs that are marked with "(u)" means that they are unitary costs, each vehicle equipped will have that cost. So to calculate the total unitary cost per vehicle the following formula is presented:

$$C = cd + sen + odev + el + \frac{ad + sl + maint + simt + oc}{N}$$

where *N* is the total amount of vehicles we have to equip (200000).

In our case:

$$C = 650 + 1850 + 930 + 300 + \frac{375000 + 2500 + 2720 + 880 + 2300}{200000}$$

As a result we have a total unitary cost of 3731.917.

This cost will be added directly to the price of the vehicle, which is an important increment of its cost. This is a young technology, and it is supposed to be expensive until the point that big companies start including it to huge number of units.

It is the decision of these companies to invest or not to invest in this technology that could expensive now but worth in the near future.

9. Conclusions

In this work the problem of autonomous driving has been addressed. The objective of this section is to summarize all the ideas presented in this thesis highlighting the most important ones.

The starting point is the nonlinear model of the vehicle and a trajectory planner. Studying the nonlinear model a simulation environment is prepared, and a set of transformations are carried out with the aim of obtaining a simpler model for control purposes. These transformations consist on linearising the model into a linear model with parameters varying.

The variant model evaluated at the bounds of its variables (vertices) building a set of IF-THEN rules that corresponds each one to a vertex of the model composing the Takagi-Sugeno fuzzy model, which will be used for control purposes as the control-oriented model (COM).

With this COM, a linear MPC is designed. Each step the scheduling variables must be evaluated and as a weighted sum of the matrix in the vertices the Takagi-Sugeno model is calculated and passed as input to the MPC. The controller computes the errors (difference between the reference given by the planner and the current states of the vehicle) from the vehicle's perspective and tries to minimize a cost function composed by these errors and the increment of the control action, resulting each iteration an optimal sequence of control actions that will drive the car to track the desired trajectory.

After designing the controller, the matrices Q and R were tuned in a way that the weights could be robust enough to manage in a proper way with different kind of circuits.

Also, a Moving Horizon Estimator (MHE) is introduced to the system in order to estimate the state that cannot be measured and the rest of the dynamic states filtering the noise in the case of applying the controller to the real car. MHE need a tuning process too that is done in a similar way than in the MPC case.

The sequence of control actions is depreciated but the first one, that is applied to the simulated-oriented model (SOM) made as from the nonlinear one. From the simulator we obtain the "real" states of the car and the performance of our controller.

Different circuits were tested and regarding to the results we can say that the controller was able to drive the car tracking the reference trajectory but with some error in some situations as the parts were there were some curves or some speed requirements variation.

9.1 Future work

Taking into account the conclusions previously described, a set of possible ways to continue the work is presented as it follows:

The first obvious next step is to test the controller with the real car instead of testing in with the controller, but maybe it is convenient to apply before this step some of the other improvements described here in order to get a better performance of the controller in simulation.

As it was said, there exists an unknown disturbance that is a variation of the friction force that actuates in our system. A good improvement will be to modify the MHE in order to estimate this disturbance and compensate it, this probably would increase the performance of the controller decreasing the error in the areas where the velocity is high.

In the case of having a better solver we could transform the linear MPC into a nonlinear one, solving the optimization problem based on the nonlinear constraints of the real car. This fact would eliminate the errors due to the linearization, however with the tools available for the project this strategy is infeasible, in the sense that is not possible to solve the nonlinear optimization problem within a feasible time interval.

Another possible improvement consists on designing a procedure that tunes the weights of the objective function in an optimal way.

Introducing the possibility that the trajectory could change along the simulation would allow combining this controller with an obstacle avoidance algorithm, such as PRM for instance. These improvements would provide the car a more realistic behavior closer to a possible real-life path.

We saw that the time that the controller took to compute the control action was more or less constant (at least bounded). The controller could be improved in order to deal with these delays and compensate them.

Bibliography

[1] T. Zhao, State feedback control interval type-2 Takagi-Sugeno fuzzy systems, 2014.

[2] Lendek, Guerra, Barbuska y D. Schutter, Stability Analysis and Nonlinear Observer Design using Takagi-Sugeno Fuzzy Models, 2014.

[3] Rastegar, Araújo y Mendes, «Online Identification of Takagi-Sugeno Fuzzy Models Based on Self-Adaptative Hierarchical Particle Swarm Optimization Algorithm,» 2017.

[4] Rodriguez-Castaño, Heredia y Ollero, «High-speed autonomous navigation system for heavy vehicles,» 2016.

[5] A.-T. Nguyen, C. Sentouh y J.-C. Popieul, «Takagi-Sugeno Model-based Steering Control for Autonomous Vehicles with Actuator Saturation».

[6] E. Alcala, «Modelling, planning and nonlinear control techniques for autonomous vehicles,» Barcelona, 2016.

[7] K. Tanaka y H. O. Wang, Fuzzy control systems, design and analysis, 2001.

[8] M. Chadli, Multiple models approach in automation, London, 2013.

[9] Q.-G. Wang, T. H. Lee, C. Lin y Y. He, LMI Approach to Analysis and Control of Takagi-Sugeno Fuzzy Systems with Time Delay, 2007.

[10] R. Tóth, Modelling and Identification of Linear Parameter-Varying Systems, 2008.

[11] J.-X. Xu, Z.-Q. Guo y T. H. Lee, Design and Implementation of a Takagi–Sugeno-Type Fuzzy Logic Controller on a Two-Wheeled Mobile Robot, 2013.

[12] P. Quiñones-Reyes, H. Benítez-Pérez, F. Cárdenas-Flores y G.-N. Fabián, Control Reconfigurable Difuso Takagi-Sugeno en Red usando Planificación EDF en xPC

Target, 2007.

[13] C. E. García, D. M. Prett y M. Morari, Model Predictive Control: Theory and Practice, 1989.

[14] D. Bertsekas, Dynamic programming and optimal control, 1995.

[15] E. L. Haseltine y J. B. Rawlings, «Critical Evaluation of Extended Kalman Filtering and Moving-Horizon Estimation,» 2005.

[16] M. Diehl, H. J. Ferreau y H. Niels, «Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation,» 2009.

[17] W. Dixon, D. M. Dawson, E. Zergeroglu y A. Behal, «Nonlinear control of wheeled mobile robots,» 2001.

[18] E. Alcala, P. V., J. Quevedo y T. Escobet, «Gain Scheduling LPV Control Scheme for the Autonomous Guidance Problem using a Dynamic Modelling Approach,» 2015.

[19] E. Alcala, P. V., J. Quevedo y T. Escobet, «Gain Scheduling LPV Control for Autonomous Vehicles including Friction Force Estimation and Compensation Mechanism,» 2015.

[20] E. Ostertag, «Mono-and Multivariable Control and Estimation: Linear, Quadratic and LMI Methods, Springer Science & Business Media,,» 2011.